**Include Line:**

#include “rtos.hpp”

**create class with rtos task**

class name : public rtos::task<>{ //every class that is using rtos items needs to inherit rtos

private:

……

public

name(const char \* name):

task(name)

{}

}

**class call in main**

auto task = name(“task name”, any, other, variables, in, your, constructor);

rtos::run();

**Timer:**

**aanmaak**

Rtos::timer name\_of\_timer;

**Constructor initialization**

Name\_of\_timer(this, “name\_of\_timer”)

**Aanroep:**

Name\_of\_timer.set(delay);

Wait(name\_of\_timer);

**clock:**

**aanmaak**

Rtos::clock name\_of\_clock;

**Constructor initialization**

Name\_of\_clock(this, delay, “name\_of\_clock”)

**Aanroep:**

Wait(name\_of\_clock);

**Flag:**

**aanmaak**

Rtos::flag name\_of\_flag;

**Constructor initialization**

Name\_of\_flag(this, “name\_of\_flag”)

**Aanroep:**

Wait(name\_of\_flag); //waiting on the flag to be set

Name\_of\_flag.set() //for setting the flag

**Writer:**

**mutex:**

**aanmaak**

**buiten class**

auto mutex\_name = rtor::mutex(“mutex\_name”) //meegeven aan de class contstructor call

**in class:**

Rtos:mutex **&** mutex\_name;

**Constructor initialization**

Mutex(mutex)

**Aanroep:**

Mutex\_name.wait(); //lock mutex

Mutex\_name.signal(); //release mutex

**Pool with mutex:**

**Aanmaak**

Mutex(zie mutex)

Rtos::pool<type> pool\_name

**Constructor initialization**

Pool\_name(“pool\_name”)//don’t forget “” signs

**Aanroep**

Mutex.wait()

Pool\_name.write(data);//or

Auto data = Pool\_name.read();

Mutex.signal()

**channel:**

**Aanmaak**

Rtos::channel<type, amount> channel\_name

**Constructor initialization**

channel\_name(This, “channel\_name”)

**Aanroep**

Channel\_name.write()

Channel\_name.read()